TLN - Radicioni

Mario Bifulco

a.a. 2022-2023

Indice

Capitolo 1

Introduzione alla semantica lessicale

Storia delle ontologie Semantica lessicale = Studio del significato delle parole e delle sue relaizoni

In particolare studia il significato dei lessemi individuali, perché hanno quel significato, come facciamo a rappresentare i lessemi e che significato si forma dalla loro combinazione (utterance = espressione)

Inizialmente il significato di termini era codificato in ontologie. Il concetto di ontologie risale ai tempi dei greci con la filosofia, ad oggi ha un ruolo in AI, linguistica computazionale e teoria dei database

Ontologia dal punto di vista metodologico è caratterizzata da una forte interdisciplinarità. Dal punto di vista arcchitetturale è centrale il ruolo che l'ontologia può avere nei sistemi informativi (ontology driven systems)

Siri era basato su un'ontologia e rispondeva facendo ricerche su essa

Lo scopo principale di un'ontologia ad oggi è strutturare la conoscenza

Filosoficamente si definisce come ciò che esiste, studio della natura e della struttura della realtà. Si tratta di un sistema strutturato di entità, organizzato in categorie e relazioni

Dal punto di vista informatico un'ontologia è un insieme di primitive rappresentazionali per modellare e rappresentare una conoscenza di dominio. Le primitive sono classi e relazioni, le definizioni includono informazioni sul significato e vincoli di applicabilità

Concettualizzazione Rappresentazione formale di un pezzo di realtà organizzato ad agenti, indipendentemente da vocabolario e occorrenze in specifiche situazioni

Situazioni diverse con vocabolari diversi possono avere una stessa rappresentgazione ontologica

Ontologie per la linguistica L'ontologia rappresenta il cosa, la semantica i riferimenti tra cose. Diversi aspetti della linguistica possono essere trattati in differenti pezzi dell'ontologia. Le basi di conoscenza sono formate da una parte terminologica e una parte di asserzioni.

Al crescere della espressività (precisione) dell'ontologia abbiamo:

1. Cataloghi
2. Glossari
3. Tassonomie
4. Thesauri
5. DB/OO scheme
6. Teoria assiomatica

Il lessico è l'insieme di parole facilmente esprimibile in un'ontologia, esistono però varie relazioni lessicali (sinonimia, iponimia, iperonimia, meronimia, olonimia, antonimia). Alcune di queste relazioni sono tipiche delle ontologie (iponimia - subclass, meronimia - has part)

Normalmente le sotto-categorie sono mutuamente esclusive, ma nel lessico spesso ci sono sovrapposizioni di significato (problema dei quasi sinonimi). Inoltre si creano buchi nel lessico, manca ogni riferimento a classi non lessicalizzate. Più il linguaggio è tecnico più è facile il "porting" in un'ontologia

Ontology design Ci sono due macro classi, entità (oggetti che mantengono una propria identità nel tempo) ed eventi (oggetti che accadono)

Per mantenere alcune proprietà in modo stnadard ci sono ontologie fondazionali, che catturano concetti indipendenti dal dominio (ad esempio DOLCE)

Dolce divide tutto in Enduranti (oggetti che esistono) e Perduranti (eventi che accadono). Tutto è suddiviso secondo la teoria delle qualità e si usa un approccio moltiplicativo (localizzazione spazio-temporale)

Capitolo 2

Knowledge representation

Inizialmente lo sviluppo AI era focalizzato su metodi generali per la risoluzione dei problemi, indipendenti dal dominio. A partire dalla metà degli anni 60 si realizza che per operare su domini complessi occorre conoscere il mondo il cui si opera

Knowledge representation studia i formalismi adatti a rappresentare le conoscenze. Tale conoscenza è esprimibile con formule logiche, ma con solo FoL non posso rappresentare i collegamenti. Per aggregare la conoscenza si usano ontologie e reti semantiche

Le reti semantiche usano le relazioni e la struttura a grafo per rappresentare i concetti. Questi formalismi soffrono comunque di problemi di espressività (non posso rappresentare implicazioni, disgiunzioni e quantificatori universali).

Inoltre bisogna bilanciare espressività e granularità, elevata granularità permette di tradurre sempre in formule per il calcolo dei predicati, tuttavia in questo modo si perde significato delle singole asserzioni

Si può sempre passare da relazioni semantiche a formule logiche, quindi non si sceglie in base al potere espressivo. Le reti risultano comode per rappresentare conoscenza gerarchica (tipica di molti contesti). Il principale vantaggio delle reti semantiche è l'ereditarietà di proprietà

Formalismo dei frame Simile alle reti semantiche per molti aspetti, serve per strutturare la conoscenza, rappresenta uno "script" di un concetto. Tuttavia manca di semantica formale

Capitolo 3

WordNet

Wordnet è un database lessicale ispirato alle teorie psico-linguistiche. I sensi sono organizzati in synset (insiemi di sinonimi) e i synset sono legati tra loro con relazioni lessicali.

Le informazioni in un dizionario standard hanno un ordinamento naturale basato sull'alfabeto quindi sinonimi e termini vicini come significato possono essere molto sparsi nella risorsa. Il problema è che non c'è altro modo di rappresentarli che sia migliore di questo

I dizionari online possono cercare molto velocemente, ma basarsi solo su una "ricerca veloce" non sfrutta le loro potenzialità

Princeton nel 1985 con un gruppo guidato da Miller crea WordNet, l'idea iniziale era creare un database in cui cercare in modo concettuale e non alfabetico

I termini sono etichettati come NOUN, VERB, ADJ e ADV. I noun sono organizzati in gerarchie, i verbi su una varietà di relazioni mentre gli aggettivi e gli avverbi in uno spazio N-dimensionale. Per i Noun sono disponibili 25 classi primitive, che sono le radici degli alberi che compongono la foresta di wordnet

- Sinonimia: relazione più importante in wordnet
- Antonimia: parola di significato opposto (meno facile da disambiguare anche per gli esseri umani), è una relazione lessicale
- Iponimia: relazione semantica tra significati (x sottoclasse di y), il suo opposto è l'iperonimia
- Meronimia: relazione parte-tutto (questo tipo di relazioni complica di molto il grafo di wordnet). Il suo opposto è la olonimia. In generale la relazione è transitiva, ma viene limitata per evitare casi limite (la maniglia è parte della casa)

Di definisce **feature funzionale** una caratteristica che le istanze hanno normalmente.

Lexical matrix La semantica lessicale vuole riconoscere le parole e le loro associazioni (concetto lessicale e ruolo nell'espressione). La matrice lessicale rappresenta le coppie (significato della parola - forma della parola), ovvero le coppie (senso, termine). I synset rappresentano una riga della matrice. I synset in wordnet hanno associata una glossa (breve descrizione del senso). Le relazioni semantiche in wordnet sono simmetriche

Assunzioni di WordNet I noun sono organizzati in relazioni che rispecchiano le teorie psico-linguistiche, gerarchia di significati. Il problema della gerarchia di WordNet è la diversa distribuzione di densità

Secondo la teoria di Rosch, 1975, c'è un livello intermedio della gerarchia in cui la maggior parte delle informazioni caratteristiche vengono catturate. Livello sopra hanno descrizioni brevi e generali, livelli sotto aggiungono poco

Verbi I verbi rappresentano la parte più importante della lingua. I verbi sfortunatamente sono più polisemici dei sostantivi, inoltre mutano molto velocemente (classe aperta). In WordNet i verbi sono organizzati in 15 file (uno per dominio semantico)

Le relazioni di iponimo tra verbi è più una troponimia (V_1 è V_2 in un particolare modo)

Capitolo 4

WSD

4.1 Similarità

Dati due termini vogliamo ottenere un punteggio che indica la loro similarità (tipicamente tra [0, 1]). Per fare ciò si può sfruttare la struttura ad albero di WordNet

$$sim_{wu-palmer}(s_1, s_2) = \frac{2 * depath(LCS)}{depth(s1) + depth(s2)}$$

Dove LCS è il primo antenato comune tra i due sensi. Questa misura di similarità assume valori in [0, 1]

$$sim_{shortest-path}(s_1, s_2) = 2 * depthMax - len(s_1, s_2)$$

Questa misura assume valori in [0, 2*depthMax]

$$sim_{leakcock-chodorow}(s_1, s_2) = -log(\frac{len(s_1, s_2)}{2 * depthMax})$$

Che assume valori in (0, log(2*depthMax + 1)]

Per ottenere i sensi avendo termini si usa un semplice approccio di massimizzazione (cerco la coppia più probabile)

$$sim(w_1, w_2) = \max_{c_1 \in s(w_1), c_2 \in s(w_2)} sim(c_1, c_2)$$

4.2 Word Sense Disambiguation

Problema aperto che si pone l'obiettivo di disambiguare il senso legato ad un termine partendo da un contesto

Per estrarre le feature rilevanti due approcci semplici ma largamente usati sono:

Collocation Considerare parole vicine (es. una a sinistra e 4 a destra)

Bag-of-words Considero tutte le parole come insieme non ordinato. L'approccio più semplice considera se una parola è presente o meno nella frase contesto tramite una vettore one hot sul vocabolario. Si può domare la dimensionalità considerando le N parole più frequenti.

Algoritmo Lesk L'algoritmo Lesk inizializza il risultato con il senso più frequente associato al termine analizzato. Poi scorre tutti i sensi del termine e calcola l'overlap tra il contesto per il dato senso e quello dela frase in input. Restituisce coem risultato il senso che massimizza tale overlap.

Il problema di questo approccio è che le frasi in input sono generalmente brevi, si possono valutare espansioni automatiche tramite parole affini o usare dei dati sense-tagged (Corpus-Lesk). Si calcola una sovrapposizione pesata, il peso è IDF $idf_i = log(\frac{Ndoc}{nd_i})$

Capitolo 5

FrameNet

Risorsa basata su due assunti teorici:

- La comprensione avviene facendo operazioni mentali su quanto si conosce già
- La conoscenza è impacchettata in strutture chiamate frame

FrameNet si basa su:

1. Caratterizzare i frame
2. Trovare le parole che rientrano nel frame
3. Sviluppare una terminologia descrittiva
4. Estrarre frasi d'esempio
5. Annotare gli esempi
6. Derivare una descrizione

I frame alla base della risorsa descrivono tipi situazionali a grana fine. Per riferirsi al frame gli sviluppatori hanno cercato una parola che "evocasse" il frame stesso.

Nei frame si parla di **Lexical Unit**, ovvero coppie (parola-senso). In wordnet LU diverse appartengono a synset diversi, in framenet appartengono direttamente a frame diversi.

Risorse come WN e FN hanno sensi altamente polverizzati, rendendo difficile l'elaborazione automatica.

Altra caratteristica dei frame è la presenza di **Frame Element**. Gli esempi in FrameNet sono spesso estratti dal British National Corpus. FrameNet ha rappresentato una risorsa chiave per NLU/NLG, MT, IE e sono state proposte molte varianti per lingue diverse dall'inglese.

Capitolo 6

N-gram Language Model

Il meccanismo alla basa di questi LM è l'uso delle probabilità condizionate. I modelli a n-grammi assegnano una probabilità a tutte le sequenze di n parole presenti nel corpus, calcolando quindi la probabilità di emissione. Il metodo naive calcola la frequenza relativa degli esempi trattati, ma questo approccio non può funzionare su larga scala, anche il web non ha abbastanza esempi per rendere questi calcoli significativi. Si applica quindi una semplificazione per considerare solo n-1 parole precedenti a quella target (ipotesi Markoviana), domando così la sparsity, dopodiché si usa la chain rule per continuare a produrre le parole successive.

Di solito per migliorare l'addestramento di questi modelli si usano tag di start e di end delle frasi. Questa procedura permette l'addestramento anche dell'inizio e della fine delle frasi con precisioni, inoltre siccome i tag non fanno parte del dizionario trattato siamo sicuri di poterli eliminare una volta generata la sequenza di parole. Le frasi prodotte seguono regole probabilistiche, in linea di principio si possono produrre sequenze di qualsiasi lunghezza, ma non c'è garanzia sulla correttezza dell'output (sia semantica che sintattica)

I modelli a N-grammi catturano in modo efficace le regolarità sintattiche o, in base ai dati, le "convenzioni". Per evitare di avere prodotti eccessivamente piccoli al posto della produttoria delle probabilità si usa la somma dei logaritmi.

Le valutazioni dei LM possono essere estrinseche (il modello si applica in un contesto e si valuta di quanto e come migliora l'esperienza), oppure intrinseca (misura standard indipendente dal contesto futuro di utilizzo). La valutazione intrinseca può essere effettuata tramite la **Perplexity**.

Perplexity La probabilità di una sequenza ($P(W)$) è:

$$\frac{1}{k}\sum_{i=1}^{k} log(LM(W))$$

$$PPL(LM; W) = e^{-P(W)}$$

A valori bassi di perplessità corrispondono frasi altamente probabili.

Gestione degli zeri Possono esserci dati non presenti nel training set che fanno degenerare le performance. Si può usare un vocabolario aperto taggando come ¡UNK¿. Le due strategie principali per gestire i tag unknown sono:

- Ricondursi ad un vocabolario chiuso scegliendo i termini significativi usando tag ¡UNK¿ per le parole non nel training set
- Creare durante l'addestramento le probabilità di ¡UNK¿ considerante come sconosciute le parole sotto una certa soglia.

Le motivazioni dietro lo smoothing sono evitare pattern con probabilità 0, che fanno degenerare i conti. Una correzione standard è quella di Laplace, dove si aggiunge 1 ad ogni conteggio.

Un altro metodo è usare n-grammi più brevi quando non si dispone della probabilità con N (decomposizione)

Capitolo 7

Semantica lessicale e rappresentazioni vettoriali

Lessico di base:

- Documento: unità di testo indicizzata
- Collezione: insieme di documenti nel sistema
- Termine: elemento della collezione (da singolo lessema a frasi)
- Interrogazione: richiesta dell'utente

Un documento in un **Vector Space Model** è espresso come vettore in cui ogni feature è detta **Term Weight** ed è la frequenza del termine nel documento. Allo stesso modo si possono creare le domande come vettori. Le feature sono dimensioni in uno spazio e i pesi sono i punti. Si possono usare misure di distanza geometrica per calcolare i risultati più probabili.

Per pesare correttamente l'apporto dei termini nel vettore si usa sia la frequenza delle parole sia la non-frequenza a livello di collezione. IDF, inverse document frequency, i termini rari sono più informativi e determinanti nel documento in cui compaiono.

$idf_i = log(\frac{N}{n_i})$ Il logaritmo serve per evitare valori troppo elevati, N è il numero di documenti e n_i è il numero di documenti in cui occorre il termine.

Metodo di Rocchio Il metodo di Rocchio permette di classificare documenti tramite una rappresentazione prototipica delle classi. Si calcola il centroide della classe dividendo la term-frequency con il numero totale di termini (vocabolario). L'idea del classificatore di Rocchio è premiare per esempi vicini (della stessa classe) e penalizzare per esempi lontani. Normalmente gli

esempi negativi sono tutti gli esempi meno i positivi, un raffinamento possibile è usare il concetto di near positive, il problema è stabilire con precisione quali siano i near-positive

Ipotesi distribuzionale Si basa sul fatto che: Due documenti sono simili se contengono parole simili e quindi riflettono la stessa rappresentazione vettoriale

Il problema del basarsi solo sui termini è con le negazioni (serve un controllo più raffinato). Inoltre alcune parole possono essere "intercambiabili"

Ipotesi di Harris, 1954, un termine è caratterizzato dai suoi vicini. Quindi l'intuizione è avere vettori che esprimono anche il contesto in cui compare la parola I contesti sono composti da finestre di parole a destra e sinistra del termine, si crea quindi un vettore di co-occorrenze. Questo tipo di vettori viene generato da dataset di grandi dimensioni, al fine di avere validità statistica. Da notare che il vettore così composto è molto sparso.

Word2Vec Per evitare vettori eccessivamente sparsi e difficili da maneggiare viene introdotta la risorsa **Word2Vec**. I vettori sono corti e densi (dimensioni da 50 a 1000), diversi ordini di grandezza in medo del vocabolario. Ogni dimensione è caratterizzata da un numero reale che può essere negativo. Intuitivamente i vettori più corti sono migliori per i task di NLP, permettono ai modelli di dover apprendere meno pesi

L'obiettivo dell'algoritmo è apprendere una rappresentazione numerica in grado di catturare l'ipotesi distribuzionale, ovvero la rete skip-gram di Word2Vec apprende a classificare (classificatore binario) se una parola è probabile compaia in un contesto

Gli embeddings sono calcolati tramite *Skip-Gram Negative Sampling*. Si esegue un algoritmo di apprendimento supervisionato. Spesso per aumentare la quantità dei dati si può usare un approccio semi-supervisionato. L'idea è usare la parola e il contesto come esempi positivi e campionare parole a cosa (non del contesto) come esempi negativi, l'apprendimento viene fatto tramite regressione logistica. Gli embeddings sono i pesi della rete

Si usa il classificatore binario per semplificare il task che sennò sarebbe troppo dispendioso dal punto di vista computazionale. Sono create due matrici, embedding e contesto, di dimensione Vocab*Embeddings, sono inizializate con valori randomici e poi si fa il training

FastText Estensione di Word2Vec che ha come obiettivo quello di gestire le parole sconosciute tramite il meccanismo dei sotto-token, le parole sono divise in sotto token di lunghezza fissa in modo da diminuire la sparsity.

Capitolo 8

Sense similarity

Le reti semantiche sono molto specializzate e spesso multilinguistiche, ma hanno difficoltà nel paragonare sensi e sono complesse da usare nei task standard.

D'altro canto gli embeddings distribuzionali sono basati sulla lingua e soffrono di meaning conglation deficiency, ma rendono facili i confronti e gli usi nei task.

La risorsa **LessLex** unisce **BabelNet** (rete semantica multilinguistica basata su wordnet e wikipedia) con **ConceptNet Numberbatch** (versione che usa i vettori di Word2Vec per esprimere i concetti di senso comune presenti in ConceptNet)

LessLex permette di fare plotting nello stesso spazio dimensionale sia i sensi che i termini, inoltre mantiene la compatibilità con tutti i dati in BabelNet

Avendo lo stesso spazio vettoriale per sensi e termini è possibile utilizzare misure di similarità più raffinate. Ad esempio la **Rank similarity** permette di pesare la similarità tra sensi sulla base del vettore terminologico

Altro approccio possibile è usare la **Neighbourhood Similarity**, questa metrica permette di aggregare i sensi in centroidi rilevanti ed evitare l'elevata polverizzazione delle risorse. Inoltre il clustering non è fatto sulla risorsa di partenza, che restando intonsa può essere usata come prima senza cambiamenti

Capitolo 9

Transformer

Il problema degli embeddings è che fanno collassare tutti i sensi in un unico embedding per il termine. I Transformer sono creati per apprendere embedding contestuali, in modo da evitare la meaning conflation.

L'input di queste reti sono gli embeddings prodotti da altri sistemi, come ad esempio Word2Vec. I Transformer sono stack di livelli (la struttura a livelli permette di apprendere rappresentazioni più significative)

Architettura Sono composti da due parti principali, *Encoder* e *Decoder*. Ogni livello dell'encoder è composto da uno strato di *Attention* e una rete Feed Forward. Al termine dei N strati l'output è passato alla fase di Decoding, composta da Self-Attention, Encoder-Decoder Attention e Feed Forward. Si abbandona la ricorrenza in favore dell'attensione

Ogni livello dell'encoder si compone di 5 operazioni:

- Si aggiunge informazione posizionale all'input
- Si generano gli embeddings (attenzione)
- Si procede con le fasi successive, non rilevanti a livello concettuale ma utili empiricamente (come *Add-Normalize*)

I vettori creati dal Transformer hanno dimensioni pari a 512

Attention Si calcolano le matrici Query, Key e Value (X per matrice dei pesi corrispondente, la matrice dei pesi è appresa nel training). La matrice X è costruita mettendo per ogni riga un embedding delle parole in input. L'attenzione è calcolata tramite la formula $Z = softmax(\frac{Q*K^T}{\sqrt{d_k}})*V$, siccome ci sono N strati avremo N attenzioni, che sono combinate tramite un'apposita matrice di pesi W^O in un unico vettore contesto.

Se K, V, Q sono prese dalla stessa parte di rete si parla di Self-Attention, quando invece K e V derivano dall'Encoder e Q viene calcolata dal Decoder si ottiene la Encoder-Decoder Attention.

Tokenization Per gestire al meglio le parole i Transformer utilizzano token di sotto-parole, tali token sono generati automaticamente tramite il metodo byte pair encoding (BPE). L'algoritmo inizializza il vocabolario usando solo i caratteri (compreso il carattere speciale di inizio e fine parole), successivamente aggrega i caratteri che più frequentemente compaiono vicini.

L'uso di sotto-parole come token rende la rete più robusta rispetto a parole sconosciute, inoltre permette di risparmiare spazio in memoria.

Positional Encoding Per aggiungere informazione posizionale nel vettore già presente si usa la funzione seno per i numeri pari e la funzione coseno per quelli dispari, a queste funzioni si passa come argomento $\frac{pos}{10000^{\frac{2*i}{d_{model}}}}$

Training Il training di queste reti è spesso diviso in una parte non supervisionata (pre-training) e un successivo fine-tuning supervisionato.

Il pre-training può essere fato tramite Masked Language Model, ovvero nascondendo alcuni token della frase (15%) oppure tramite Next Sentence Prediction, in cui si chiede direttamente la frase successiva a quella fornita

Problemi dei Transformer Nonostante l'utilizzo in larga scala il meccanismo dell'attenzione ha tempo di esecuzione quadratico

L'encoding di posizioni assolute non è ottimale, architetture moderne usano posizioni relative

Capitolo 10

Riassunto automatico

BabelNet nasce dall'unione di WordNet e Wikipedia. La struttura di BabelNet è quella di un grafo etichettato e direzionato. Il grafo è stato costruito partendo da WordNet e espandendolo tramite le relazioni di hyperlink nelle pagine di Wikipedia.

Questo meccanismo ha portato ad avere un grafo incredibilmente vasto e ricco, ma non vettoriale. La risorsa **NASARI** rappresenta in uno spazio vettoriale la distribuzione semantica di BabelNet.

I vettori NASARI sono di due tipi:

- LEX, ovvero che usano i lemmi come atomi
- UNIFICATI, che usano i concetti come atomi

Nella versione unificata prima viene eseguito un clustering basato su ipernonimi, dopo si esegue un raffinamento sugli iponimi che non vengono inclusi con il meccanismo precedente.

Summarization Uno dei task comuni in NLP è il riassunto automatico, ovvero restituire versioni più brevi di un testo con le informazioni rilevanti. Un buon riassunto deve essere **indicativo** del testo originale, **informativo** e **critico**. In generale i riassunti si possono dividere in **extrattivi**, ovvero si estraggono i punti salienti o **astrattivi**, ovvero rigenerano il testo da capo. I riassunti devono focalizzarsi anche sul target di riferimento da cui verranno letti.

I riassunti si possono produrre con metodi **shallow**, ovvero basati sul recuperare le informazioni con analisi sintattiche (per produrre estratti), oppure **deeper**, in cui gli abstract sono prodotti con analisi semantiche (spesso serve conoscenza a priori)

Inoltre la tipologia di riassunto cambia se vogliamo riassumere un singolo testo o un insieme di testi.

Per capire cosa è rilevante in un testo ci si può basare su:

- Posizione nel testo: le parti rilevanti di solito sono a inizio testo (85%) e fine testo (7%)
- Metodo dei titoli: mediamente un titolo indica il contenuto del testo (non vale per romanzi o blog)
- Optimum Position Policy: trovare parti rilevanti in base a termini e posizione nel testo
- Cue phrases: alcune frasi con struttura standard sono molto indicative (si possono usare parole bonus e stigma)
- Cohesion-based: le frasi importanti sono quelle più connesse (es. co-occorrenze)

I sistemi che riassumono devono selezionare le frasi rilevanti, organizzarle e produrre il nuovo testo. Il più semplice algoritmo non supervisionato seleziona le frasi in base alle parole più informative.

Dopo aver scelto un metodo per creare il vettore contestuale (ad esempio tramite NASARI) si può calcolare la similarità tra i paragrafi e il contesto con la square-root Weighted Overlap.

$$sim(w_1, w_2) = \max_{v_1 \in C_{w_1}, v_2 \in C_{w_2}} \sqrt{\frac{\sum_{q \in O}(rank(q, v_1) + rank(q, v_2))^{-1}}{\sum_{i=1}^{|O|}(2i)^{-1}}}$$

Dove O è l'overlap tra i vettori contestuali.